

CLAIMS

1. (currently amended) A method for executing a target application on a host processor comprising:

interpreting each of a sequence of target instructions and executing the interpreted target instructions;

dynamically translating into host instructions each of a sequence of target instructions, wherein the translating is performed when the number of times a sequence of target instructions is executed exceeds a preset count;

storing the translated host instructions wherein a plurality of additional registers are used during execution for holding the official state of a target processor;

responding to an exception during execution of a stored translated instruction by rolling back to a previous point in execution at which correct state of a target processor is known; and

interpreting each target instruction in order from the point in execution at which correct state of a target processor is known.

2. (previously presented) The method of Claim 1 which further comprises:

collecting statistics regarding the execution of sequences of instructions which are interpreted.

3. (currently amended) A method for executing a target application on a host processor comprising the steps of:

interpreting sequentially each of a sequence of host instructions and executing the interpreted host instructions representing each target instruction of the target application;

performing a dynamic translation process when the number of times a host instruction is executed exceeds a preset count wherein a plurality of additional registers are used during execution for holding the official state of a target processor;

responding to an exception during execution of translated host instructions representing a target instruction by returning to a previous point in execution of the target application at which correct state of a target processor is known; and

thereafter executing host instructions by interpretation of the target instruction until the point of the exception.

4. (previously presented) A method as claimed in Claim 3 which further comprises collecting statistics regarding the execution of sequences of target instructions which are executed.

5. (previously presented) A method as claimed in Claim 4 in which the statistics include the number of times the sequence of target instructions have executed.

6. (previously presented) A method as claimed in Claim 4 in which the statistics include address of an instruction to which a target instruction including a branch operation branches.

7. (previously presented) A method as claimed in Claim 4 in which the statistics include a likelihood of a branch being taken.

8. (currently amended) A computer-readable medium encoded with a computer program for implementing a system for executing a target application designed for execution on a target processor on a host processor having an instruction set different than that of the target processor comprising:

means for interpreting a sequence of target instructions and executing each the interpreted target instructions;

means for dynamically translating sequences of target instructions and storing each translated sequence of instructions, wherein the translating is performed when the number of times a sequence of target instructions is executed exceeds a preset count;

means for storing each translated sequence of instructions wherein a plurality of additional registers are used during execution for holding the official state of said target processor,

means for responding to an exception during execution of a stored translated instruction by rolling back to a previous point in execution at which correct state of a target processor is known, and

means for interpreting each target instruction in order from a point in execution at which correct state of a target processor is known through the target instruction causing the exception.

9. (previously presented) A system as claimed in Claim 8 in which the means for interpreting is an interpreter software executing on the host processor, and

the means for translating is dynamic translation software executing on the host processor.

Claim 10. (cancelled)

Claim 11. (cancelled)

Claim 12. (cancelled)

Claim 13. (cancelled)

14. (cancelled)

15. (previously presented) The method of Claim 1, further comprising:
speculatively translating target instructions into host instructions
based on a likelihood of a branch being taken.

16. (previously presented) The method of Claim 1, further comprising:
collecting statistics regarding the execution of sequences of target
instructions which are executed.

17. (previously presented) The method of Claim 16, wherein the
statistics include the number of times a branch to target instructions have
executed.

18. (previously presented) The method of Claim 17, further
comprising:
speculatively translating target instructions into host instructions
based on a likelihood of a branch being taken.

19. (previously presented) The system of Claim 8, further comprising:
collecting statistics including the number of times a branch of target
instructions have executed.

20. (previously presented) The system of Claim 19, further comprising:

speculatively translating target instructions into host instructions based on a likelihood of a branch being taken.

21. (currently amended) A method for executing a target application on a host processor comprising:

interpreting each of a sequence of target instructions and executing the interpreted target instructions;

dynamically translating into host instructions each of a sequence of target instructions, wherein the translating is performed when the number of times a sequence of target instructions is executed exceeds a preset count;

storing the translated host instructions wherein a plurality of additional registers are used during execution for holding the official state of a target processor and wherein a buffer stores working memory state changes and official memory state changes;

responding to an exception during execution of a stored translated instruction by rolling back to a previous point in execution at which correct state of a target processor is known; and

interpreting each target instruction in order from the point in execution at which correct state of a target processor is known.